NUDTSNA at TREC 2015 Microblog Track:
A Live Retrieval System Framework for Social Network based on Semantic Expansion and Quality Model

Xiang Zhu, Jiuming Huang, Sheng Zhu, Ming Chen, Chenlu Zhang,
Li Zhenzhen, Huang Dongchuan, Zhao Chengliang, Aiping Li, Yan Jia
School of Computer
National University of Defense Technology
Changsha, Hunan 410073
Email: zhuxiang@nudt.edu.cn, jiuming.huang@qq.com, zhusheng002@126.com, cmkdcmkd@126.com,
zhangchenlu.com@163.com, lzz1217@yeah.net, polishgt@163.com,
zhaochengliang1991@163.com, apli1974@gmail.com, jiayanjy@vip.sina.com

Abstract—**This paper describe our approaches to real-time filtering task in the TREC 2015 Microblog track, including push notifications on a mobile phone task and periodic email digest task. In the push notifications on a mobile phone task, we apply a recommendation framework with rank algorithm and dynamic threshold adjustment which utilizes both semantic content and quality of a tweet. External information extracted from Google search engine and word2vec model based on existing corpus are well incorporated to enhance the understanding of a tweet's or a profile's interest. In the email digest task, based on the candidate tweets retrieved from the first task, we calculate the score of a tweet considering semantic features and quality features, all the tweets classified into a topic are ranked by our key word bool logistic model.**

1. Introduction

Information retrieval and recommendation in online social network has attracted increasing attention with development of social network services. To explore user's interests and boost retrieval and recommendation performance in real-time environment, TREC first introduced real-time task in 2011 [1], which is addressing a real-time adhoc search task. The information a user wishes to see is represented by a query, systems should respond to a query by providing a list of relevant tweets ordered by time, starting from the query is issued. In other words, systems should feed users with the most recently and relevant tweets. The Microblog Track in 2015 is a real-time filtering task, the goal of the real-time filtering task is to explore technologies for monitoring a stream of social media posts with respect to a user's interest profile. Different from a typical ad hoc query, there is not an actual information need. Instead, the goal is for a system to push interesting content to a user. The notion of what's interesting is considered in two concrete task models, push notification on a mobile phone as Scenario A and periodic email digest as Scenario B. In Scenario A, content identified as interesting by a system based on user's interest profile might be shown to the user through mobile phone notification. Under that circumstances, such notifications should be triggered a relatively short time after the content is generated. In Scenario B, content calculated as interesting by a system based on user's interest profile might be aggregated into an email form that periodically sent to a user. In that case, a user could read a longer story about the contents.

In the Scenario A, we apply a recommendation framework with rank algorithm and dynamic threshold adjustment. Semantic features and quality features are extracted to achieve good retrieval and recommendation performance in social media. For semantic features, we utilize different retrieval models, such as TFIDF, BM25, key word bool logic model, to calculate the relevance score of a given profile and a tweet. In order to enhance the performance of semantic features and ease the shortcomings of bag-of-words(BoW) model, we take advantage of word2vec model [2] [3] [4] based on existing corpus, such as Wikipedia, KnowItAll [5], Freebase [6], Probase [7]. In order to expand semantic features of profiles, we also use Google search engine to acquire external information. We use abstract text of retrieval results to better understand the user's interests. For the quality features, we utilize several quality features extract from a tweet, such as the user who post the tweet, the number of repost, the number of comment, the number of URL, the number of hashtags, the number of meaningful words, the length of a tweet, etc. Topics for TREC 2013 Microblog track [8] are used for model training. With the artificial labeled data, we obtain our quality model. Finally, we combine semantic features and quality features to evaluate a tweet comprehensively. According to dynamic threshold adjustment, a tweet is decided to push or not by our system.

The candidate tweets identified in Scenario A are used as the input source of the task in Scenario B. We calculate the score of a tweet considering semantic features and quality features. The semantic features are used to classified a tweet into a topic or drop it if it does not match any topics and the tweet classified into a topic will get a semantic score. Then quality features are utilized to evaluate the importance and authority of a tweet. By the quality model we obtained, we could get a quality score of a tweet. With a rank framework, the tweets classified into a same topic can be ranked. The top k tweets will be pushed to the user who are interested in as a digest.

The remainder of the paper is organized as follows, we first propose our approach for push notifications on a mobile phone task in Section 2. In Section 3, we describe our system for periodic email digest task in detail. Section 4 presents our experimental results and analysis. At last we conclude our paper in Section 5.

2. Push Notification on a mobile phone Task

In this section, we first introduce our system architecture for push notifications on a mobile phone task. Then, the recommendation framework are demonstrated in detail. At last, all the components of the system are presented.

2.1. System Overview

It is a real-time job in this year's Microblog track that teams listen to the twitter stream [9] via official common API. In this section, we briefly discuss the architecture of our system, which is shown in Figure.1. As depicted in the figure, we can see our system mainly contains four components as follow,

1) **Feature Extraction Component**, which extract features from twitter stream based on TREC-API[1] and profiles provided by the official. Before feature extracting, data preprocessing and data filtering is implemented to get rid of unnecessary data. For twitter stream, we extract semantic features and social attributes. For profiles, we extract key words as our basic features.
2) **Feature Representation Component**, which represents and expands semantic features by several techniques. Information of a tweet and profiles are enriched by this component.
3) **Candidate Generation Component**, which classifies tweets to the optimal profile based on semantic features and quality features by a key word bool logic model.
4) **Scoring and Pushing Component**, which ranks tweets candidates in different profiles with the final score and makes threshold adjustment based on historical data over time.

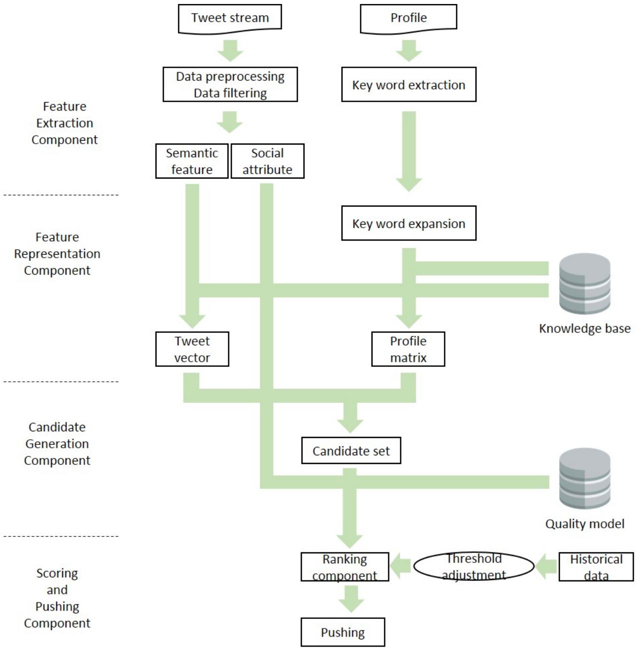

Figure 1. System Architecture Framework

2.2. Feature Extraction Component

Twitter stream we listen to is during the evaluation time according to the official[2], and it lasts ten days. After obtaining twitter stream, we adopt preprocessing and filtering to reduce the tweets we need to process. The preprocessing and filtering utilized on tweet stream are as follow,

- **Non-English Filtering**, we abandoned the non-English tweets by a language detector with infinity gram, named ldig [10]. This tool kit is a prototype for short message service with 99.1% accuracy for 17 languages[3]. By the way, we also use a method based on encoding set of characters to process tweets consist of both English characters and non-English characters. We only keep the tweet in which English characters is the vast majority with a threshold value.
- **Redundant Retweet Elimination**, we only keep one tweet and eliminate other tweets retweeted the same tweet by the retweet id information according to official requirements.

Then semantic features and social attributes are extracted from tweets. For semantic features, we selected nouns and verbs in tweet text. So semantic features of a tweet is represented as Equation.1,

$$T = \{t_1, t_2, ..., t_n\} \tag{1}$$

T represents a tweet and t_i stands for a key word in tweet text. The social attributes are extracted from structured data

[1] https://github.com/lintool/twitter-tools

[2] https://github.com/lintool/twitter-tools/wiki/TREC-2015-Track-Guidelines

[3] https://github.com/shuyo/ldig

in a tweet. A tweet is structured as JSON format, it is convenient to get social attributes we need, such as the user who post the tweet, the number of repost, the number of comment, the number of URL, the number of hashtags, the number of meaningful words, the length of a tweet, etc.

For profiles, we extract the nouns and verbs from *title*, *desc* and *narr* field. We use a key word bool logic model to express the information of a profile as follow,

$$P = \{tid : xxx, keyword : \{0 : p_1||p_2, 1 : p_3\&\&p_4\}\} \quad (2)$$

P represents a profile, tid stands for a topic id of a profile. The keyword field contains two fields, 0 for words that unnecessary but could increase the semantic score and 1 for words that need to be included. Symbol || means *or* logic and symbol && stands for *and* logic. So it means p_3 and p_4 need to be included and p_1 or p_2 is optional for the profile of which topic id is xxx. In this section, we extract the features and store them by format.

2.3. Feature Representation Component

After extracting the semantic features, we need to represent those features in a proper format so that it is convenient to calculate the relevance between tweets and profiles. For profiles, the key words extract from the files offered by the official is not enough to improve the performance because short text retrieval suffers severely from vocabulary mismatch problem. Terms overlapping between profiles and tweets are relatively small. Semantic expansion methods can be leveraged to enhance the retrieval performance. In this section, we introduce several semantic expansion methods to boost the performance.

There are two kinds of semantic expansion methods, knowledge repository based and search engine based. For profiles, we use Google search engine API to expand information about the profiles. The *title* field is used as a query for searching and the abstract text information of top 50 retrieval results are collected for each profile. Abstract text is treated as a document, each document contains several terms. After gathering all the documents, we use TFIDF algorithm to calculate TFIDF value of each term for all the profiles. The top k terms of each profiles are added to key word table in Equation.2 to expand the information.

Due to the vocabulary mismatch problem, vector model is utilized to process the semantic features. The word2vec technique is used to vectorization for the key words and *gensim*[4] tool is used in this paper. The training corpus we used is acquired from *wikipedia* English corpus. A word2vec knowledge base are trained by *gensim* tool using *wikipedia* English corpus. Tweets and Profiles can be represented by word2vec knowledge base as follow,

$$T_{vec} = (t_1, t_2, ..., t_n)^T \quad (3)$$

In Equation.3, n is the dimensions in *gensim* tool, generally set to 200 or 400. The profiles can be demonstrated as a matrix as follow,

$$P_{mat} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} \end{bmatrix} \quad (4)$$

In Equation.4, n is same as in Equation.3 and m stands for the number of profiles. A row $(p_{i1}, ..., p_{in})$ in the matrix stands for the normalized center vector of a profile by all the key words. After the procedure above, the semantic features of tweets and profiles are well represented.

2.4. Candidate Generation Component

In this section, we classify tweets into the most relevant profile or drop it directly if it does not match any profile and generate candidates based on semantic features in section 2.3. Firstly semantic features are utilized based on Equation.3 and Equation.4 as follow,

$$C = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} = P_{mat} \cdot T_{vec} \quad (5)$$

Then, the profile which has the maximum value and the terms in tweet satisfy the bool logic in Equation.2 will be choose as candidate. The semantic score c_i is recorded simultaneously. We used two kinds semantic score to evaluate the relevance between tweets and profiles as follow,

- **TFIDF Score**, which calculates the cosine similarity between a tweet and a profile in vector space model with TFIDF weight of terms. Vector space model is a model which represents a document as a vector. Tweets and profiles can be expressed as vectors,

$$\vec{T} = (t_1, t_2, ..., t_n) \quad (6)$$

$$\vec{P} = (p_1, p_2, ..., p_n) \quad (7)$$

The TFIDF method use term weight and cosine similarity metric to evaluate the relevance between a tweet and a profile. Cosine similarity metric is defined as follow,

$$Sim = \frac{\vec{T} \cdot \vec{P}}{||\vec{T}|| \cdot ||\vec{P}||} \quad (8)$$

- **BM25 Score**, which utilizes the Okapi BM25 weighting function to measure the semantic relevance between a tweet and a profile. Okapi BM25 model is a bag of words model that rank documents based on the query terms appearing in each documents. The similarity between a document and a query is defined as Equation.9, where D represents a document, Q stands for a query. $f(q_i, D)$ is q_i's term frequency in document D, $|D|$ is the length of the document D in words, $avgdl$ is the average

[4] http://radimrehurek.com/gensim/index.html

document length of all the documents to process and k_1 and b is adjustable parameters.

$$Sim = \sum_{q_i \in Q} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{9}$$

Social attributes extracted in section 2.1 are used to train quality model. We label a tweet with a score from 0 to 1 artificially based on its quality. If the tweet provides more information and written more elaborately, it will get higher quality score. The model we use is logistic regression model in machine learning tool *weka*[5]. Then, the semantic score and quality score are utilized to evaluate the relevance and quality of a tweet for a certain profile. Based on the assumption that users prefer those tweets related to the profile and popular in social media, we consider social attributes as follow,

- **User follower Count**(*FollowerCnt*), which represents the number of followers of the user who post the tweet. The user whose followers count is high would be a famous user in social media and will post high quality tweet with a large probability.
- **User status Count**(*StatusCnt*), which represents the number of status of the user who post the tweet. The status count indicates the vitality of a user in social media. A energetic user will post higher quality tweet than others.
- **Retweet Count**(*RetweetCnt*), which represents the times a tweet is retweeted. The larger retweet count is, the more popular a tweet is in social media.
- **Retweet Level**(*RetweetLvl*), we use logarithm to measure retweet count to retweet level.
- **Collect Count**(*CollectCnt*), which represents the number of people who like it. People can collect a tweet or star a tweet if the tweet is attractive.
- **Word Count**(*WordCnt*), which calculates the number of words in a tweet without stop words. Generally, informative and high quality tweets may be longer than others.
- **Character Count**(*CharCnt*), which calculates the number of characters of a tweet without stop words.
- **Short Url Count**(*UrlCnt*), which represents the number of short url count of a tweet. Informative tweet and news will give a short url at the end of a tweet in general.

2.5. Scoring and Pushing Component

By the semantic features and social attributes, we got two score, the semantic score c_i in Equation.5 and the quality score q_i. Both value of them are from 0 to 1. The finally score we measured for a tweet to a profile is as follow, where s_i stands for the final score.

$$s_i = c_i \cdot q_i \tag{10}$$

When a candidate is added to the pushing queue, it is ranked by the final score s_i. If a tweet is relevant and important to a profile, it is necessary to push it to the users who are interested in. But there is a limit in Scenario A that ten tweets could be pushed to a profile at most in one day and the gain will decrease over time. So it is a constraint satisfaction problem we need to handle. We used a dynamic threshold adjustment to make sure there are enough tweets for a profile and each tweet with a high score during one day. With a recently historical data of the tweets for a profile, we can get the highest final score s_{max}. We make a piecewise function for the threshold as Equation.11,

$$threshold = \begin{cases} (0.9 - d) \cdot s_{\max} & d < 0.4 \\ 0.5 \cdot s_{\max} & d \geq 0.4 \end{cases} \tag{11}$$

where d stands for decay value and $d = c \cdot floor(t/2)$. c is decay coefficient which we set to 0.05 in our system, t is the hour in a day from 0 to 24. If a tweet's final score s_i exceed the $threshold$ at that time, it will be pushed immediately.

As described above, the live push algorithm based on semantic features and social attributes are summarized in Algorithm.1, the program won't stop until R is full for each profile with 10 tweets or Ts is exhausted in a day. $threshold$ will automatically adjust over time.

Algorithm 1 Live Push Algorithm

```
Require:
    Twitter stream $Ts = \{ts_1, ts_2, ..., ts_k\}$
    Profile document set $P = \{P_1, P_2, ..., P_m\}$
Ensure:
    Retrieval Set $R = \{R_1, R_2, ..., R_m\}$ for each profile $P_i$
    is full or $Ts = \emptyset$
$P_{mat}$ = matrix($P$)
while $R$ is not full and time is not up do
    $T_i = pop(Ts)$
    preprocess($t_i$)
    $T_{vec}$ = vectorization($t_i$)
    $C = P_{mat} \cdot T_{vec}$
    $c_j = \max(C)$
    $s_j = c_j \cdot q_j$
    if $s_j > threshold$ then
        $R_j = R_j \cup ts_i$
    end if
end while
```

3. Periodic Email Digest Task

In periodic email digest task, we need to collect a batch of up to top 100 interesting tweets for each profile during one day and deliver those information to the particular profile after the day ends. It is expected that the system will complete that mission in a relatively short amount of time. The system framework used in scenario B is same as in scenario A as Figure.1 , except threshold adjustment component. All the tweets are classified into one profile or drop it if it does not match any profile, then the candidates

[5] http://www.cs.waikato.ac.nz/ml/weka/

are ranked by final score s based on semantic features and social attributes.

To supply diverse information for a particular profile, we utilized two kinds of techniques to eliminate redundant tweets.

- **Redundancy Removal based on Id**, which utilized the tweet's id to identify a tweet. If a tweet is original, we record the id of original tweet. If it is a tweet reposting another tweet, we record the id of the reposted tweet's id. It could decrease the tweet reposting a popular tweet.
- **Simhash** [11] [12], which is a popular method to handle web page redundancy. It turns a document into a fingerprint, called simhash code. The closer hamming distances between two documents is, the more similar they are. The simhash code is calculated as follow,

  $$Sim_{code} = sign(\sum_{i=1}^{n} w_i \cdot c_i) \tag{12}$$

  where w_i is the weight of term i and c_i is the hash code of term i, $sign$ is symbol function that make positive to 1 and negative to 0 for every bit in the code.

Our daily retrieval algorithm can be described as Algorithm.2

Algorithm 2 Daily Retrieval Algorithm

```
Require:
    Twitter retrieval set $Tr = \{Tr_1, Tr_2, ..., Tr_m\}$ based on scenario A
Ensure:
    Daily retrieval Set $Dr = \{Dr_1, Dr_2, ..., Dr_m\}$
for $Tr_i \in Tr$ do
    while $|Dr_i| \leq N$ and $Tr_i \neq \emptyset$ do
        $t_{max} = \max(Tr_i)$
        $Tr_i = Tr_i - t_{max}$
        if $t_{max}$ not in $Dr_i$ then
            $Dr_i = Dr_i \cup t_{\max}$
        end if
    end while
end for
```

where Tr is daily candidates for m profiles acquired in scenario A. For each profile, we iteratively get the most interesting tweet from candidate set and drop the redundant tweet. At last, we get the daily retrieval set Dr.

4. Result and Analysis

The evaluation of TREC 2015 Microblog track lasts 10 days from Monday, July 20, 2015, 00:00:00 UTC to July 29, 2015, 23:59:59 UTC. It consists of 225 interest profiles, which the participants will be responsible for tracking. During the evaluation time, participants will listen to tweet stream continuously and deal with every tweet. After the evaluation period, based on post hoc analysis, NIST will select a set of approximately 50 topics that will actually be assessed.

TABLE 1. RESULTS IN SCENARIO A

	ELG	nCG
$SNACSA$	**0.3086**	**0.3349**
$SNACS\_LA$	0.2863	0.2974
$summaryA$	0.4623	0.4846

TABLE 2. RESULTS IN SCENARIO B

	nDCG
$SNACS$	0.3345
$SNACS\_LB$	**0.3670**
$summaryB$	0.5014

There are some metrics to evaluate the performance of a system. In scenario A, the first metric is expected latency-discounted gain (ELG) from the temporal summarization track, the ELG score is depicted as Equation.13

$$ELG = (1/|Tr|) \cdot \sum_{i} gain(Tr_i) \tag{13}$$

where Tr is the returned tweet sets, $gain()$ is the score function for a tweet. Not interesting, spam/junk tweets receive a gain of 0, somewhat interesting tweets receive a gain of 0.5, very interesting tweets receive a gain of 1.0. In addition, a latency penalty is applied to all tweets, the latency penalty is computed as $\max(0, (100 - delay)/100)$, where the delay is the time elapsed(in minutes, rounded down) between the tweet creation time and the putative time the tweet is delivered. The secondary metric is normalized cumulative gain (nCG), which is depicted as Equation.14

$$nCG = (1/Z) \cdot \sum_{i} gain(Tr_i) \tag{14}$$

where Z is the maximum possible gain (given the 10 tweets per day limit).

In scenario B, for each topic, the list of tweets returned per day will be treated as a ranked list and from this nDCG@k will be computed. The score of a topic is the average of the nDCG@k scores across all days in the evaluation period. The score of the run is the average over all topics.

The results our system get is listed in Table.1 and Table.2,

$SNACSA$ and $SNACS$ are the results pair that only use the words in tweets to generate profiles in Equation.2. $SNACS\_LA$ and $SNACS\_LB$ are the results pair that using search engines to expand to generate profiles. The $summaryA$ and $summaryB$ is the average score of the highest score of every topics. Non-expand algorithm gets higher ELG and nCG, however expand algorithm gets higher nDCG.

Figure.2 is the ELG vs. nCG pair of participants' runs, Figure.3 is the ELG distribution in different topics, Figure.4 is the nCG distribution in different topics and Figure.5 is the nDCG distribution in different topics. We can see our system is close to the max results in summaryA and summaryB among most topics, our algorithm is verified to be effective and efficient.

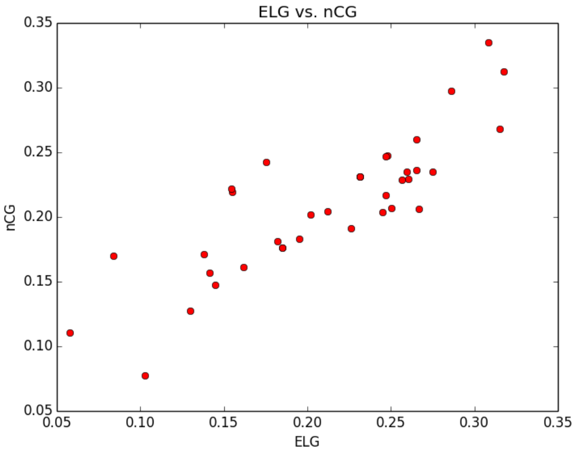

Figure 2. ELG vs. nCG

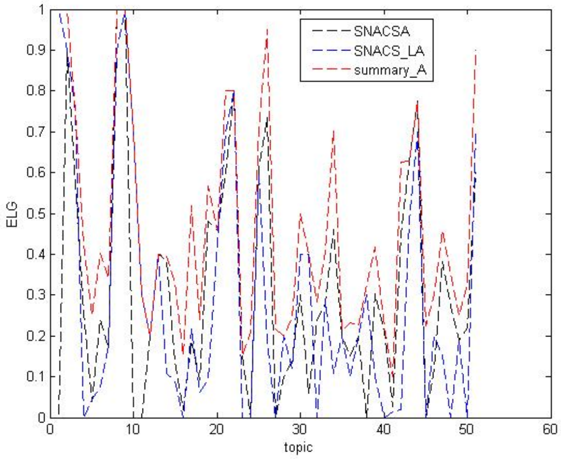

Figure 3. ELG distribution in different topics

5. Conclusion

In this paper, we present our system architecture framework and algorithms for TREC 2015 Microblog track. In the push notification on a mobile phone task, we apply a recommendation framework with rank algorithm and dynamic threshold adjustment which utilize not only semantic features but also social attributes in social media. In periodic email digest task, we calculate the score of a tweet considering semantic features and quality features, then we rank the tweets take the redundance into consideration. Experimental results show our effectiveness and efficiency of our system in both tasks.

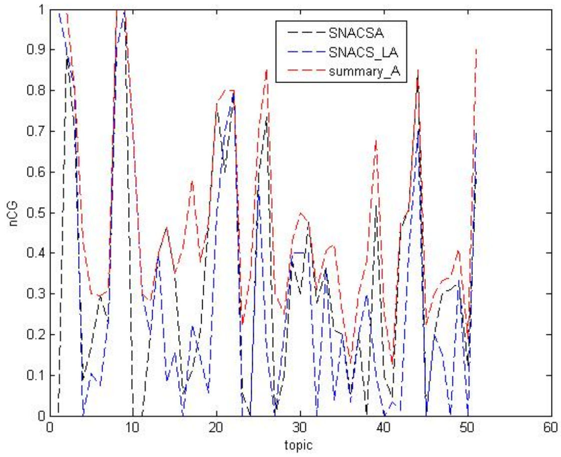

Figure 4. nCG distribution in different topics

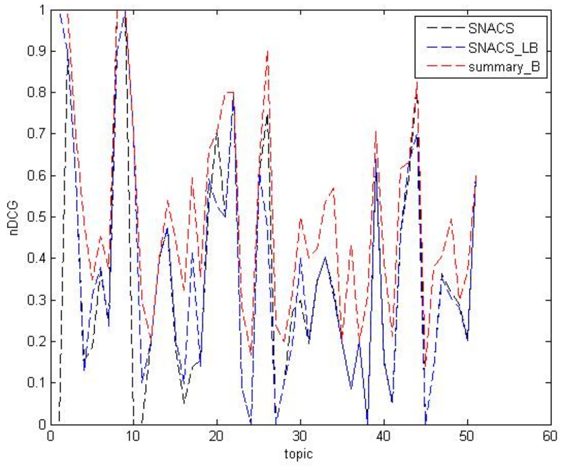

Figure 5. nDCG distribution in different topics

Acknowledgments

Sponsored by National Key fundamental Research and Development Program No.2013CB329601 and National Natural Science Foundation of China No.61372191.

References

[1] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff, "Overview of the trec-2011 microblog track," in *Proceeddings of the 20th Text REtrieval Conference (TREC 2011)*, 2011.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[4] T. Mikolov, W.-t. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations.” in *HLT-NAACL*, 2013, pp. 746–751.

[5] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, “Web-scale information extraction in knowitall:(preliminary results),” in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 100–110.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 1247–1250.

[7] W. Wu, H. Li, H. Wang, and K. Q. Zhu, “Probase: A probabilistic taxonomy for text understanding,” in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 481–492.

[8] J. Lin and M. Efron, “Overview of the trec-2013 microblog track,” in *Proceedings of TREC*, vol. 2013, 2013.

[9] J. H. Paik and J. Lin, “Do multiple listeners to the public twitter sample stream receive the same tweets?”

[10] C. L. F. F. R. Qiang and Y. F. J. Yang, “Pkuicst at trec 2014 microblog track: Feature extraction for effective microblog search and adaptive clustering algorithms for ttg.”

[11] G. S. Manku, A. Jain, and A. Das Sarma, “Detecting near-duplicates for web crawling,” in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 141–150.

[12] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 380–388.